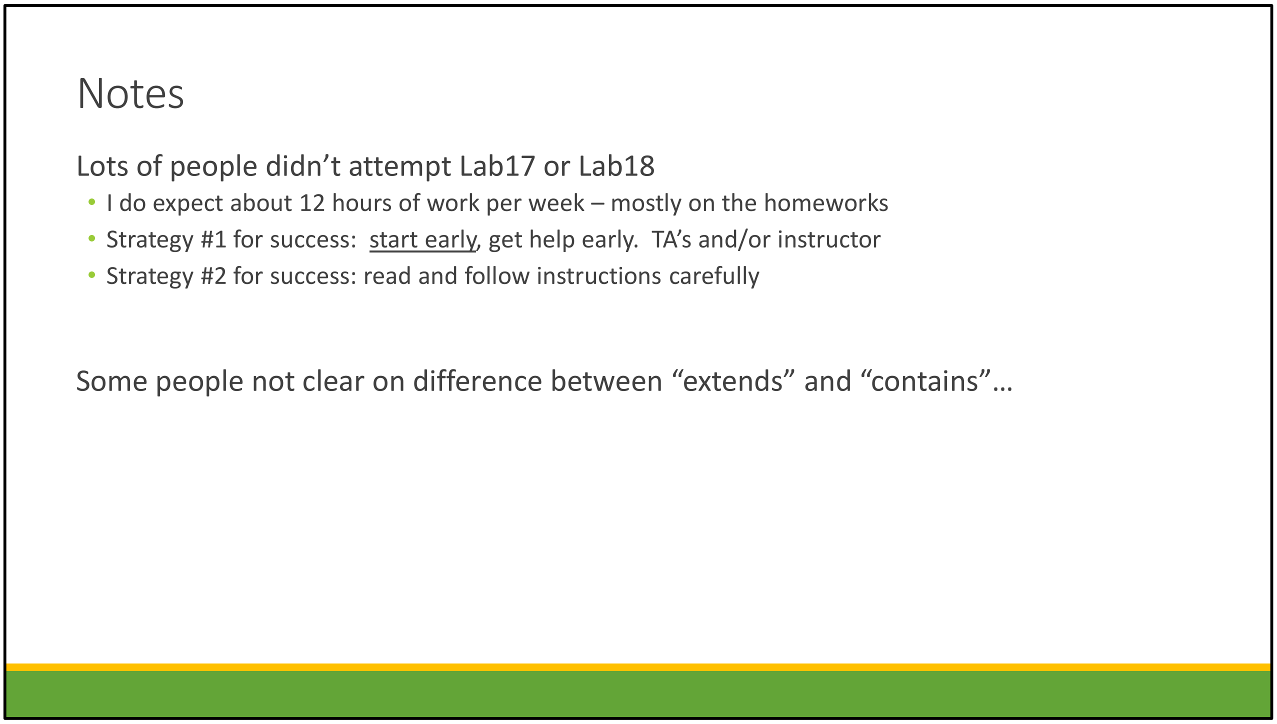


Notes

Lots of people didn't attempt Lab17 or Lab18

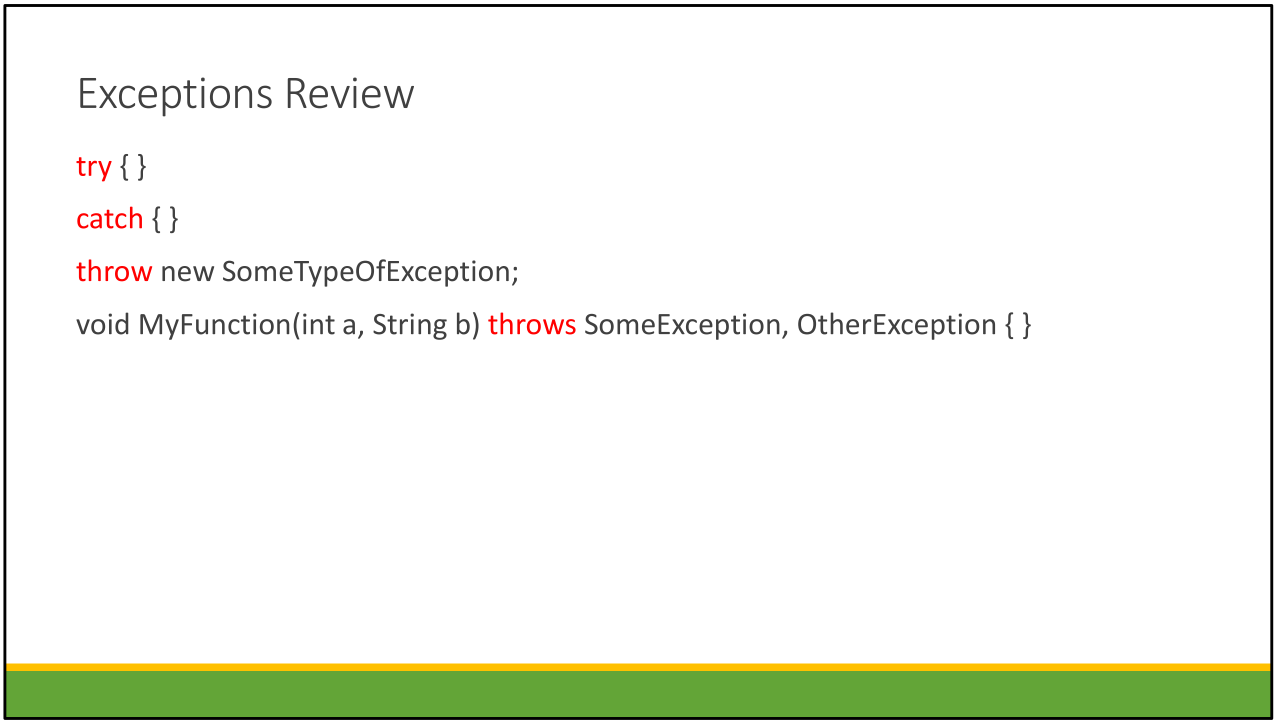
- I do expect about 12 hours of work per week – mostly on the homeworks
- Strategy #1 for success: start early, get help early. TA's and/or instructor
- Strategy #2 for success: read and follow instructions carefully

Some people not clear on difference between "extends" and "contains"…

PRINT OUT INSTRUCTIONS. CROSS OUT EACH DETAIL AS YOU code IT. HIGHLIGHT IT AFTER YOU test IT SUCCESSFULLY.

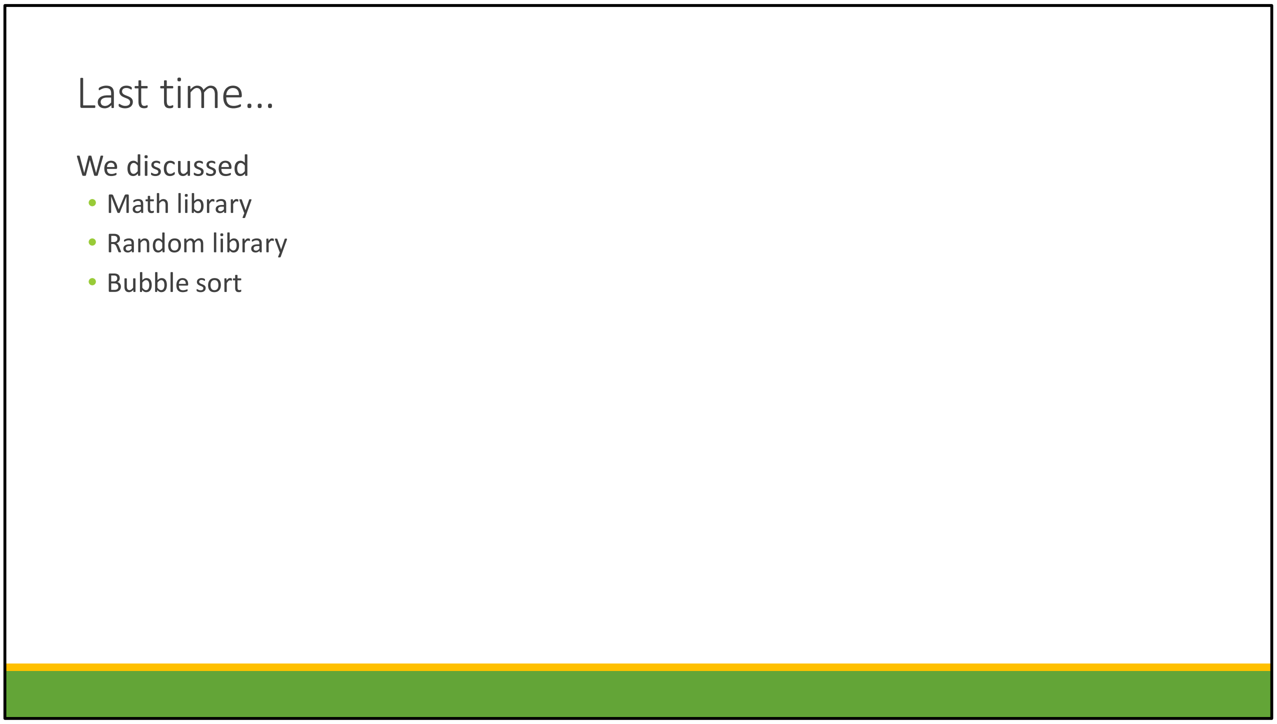
Review my soln for Lab17

Project02: no mysteries to figure out as in Wordle or Blackjack. But a lot of instructions to follow accurately.



Exceptions Review

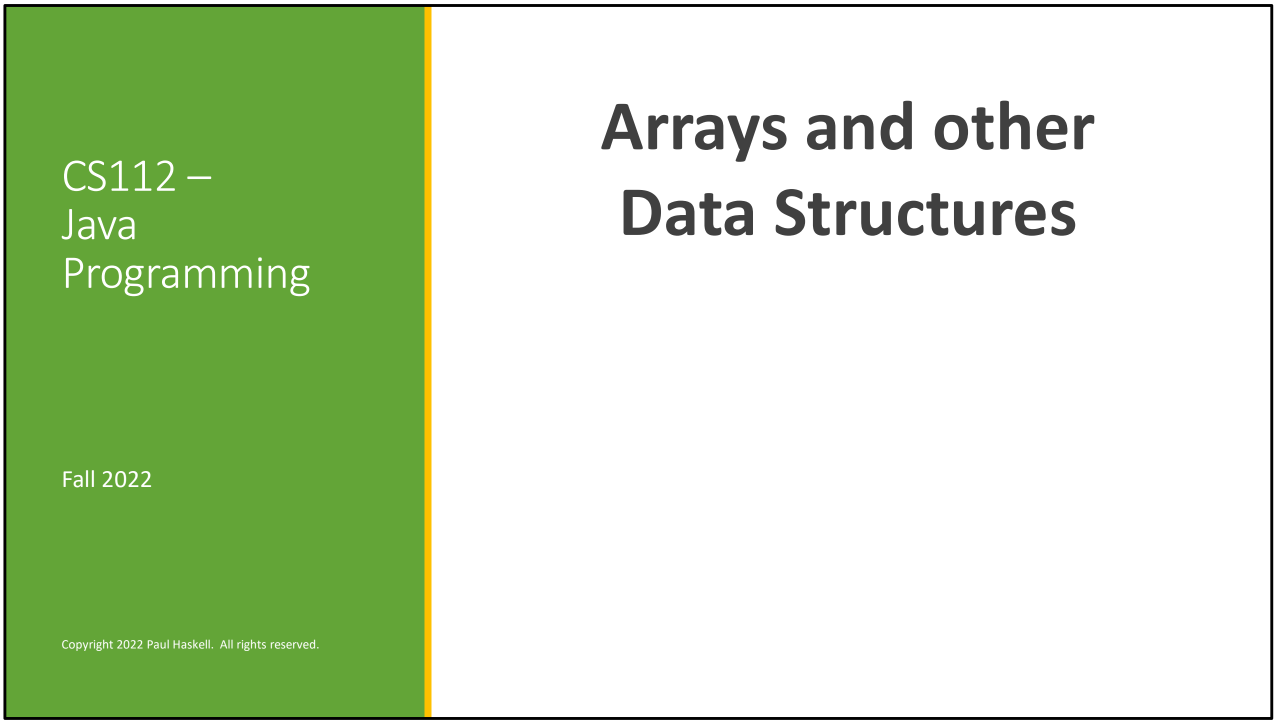
```
try { }
catch { }
throw new SomeTypeOfException;
void MyFunction(int a, String b) throws SomeException, OtherException { }
```



Last time…

We discussed

- Math library
- Random library
- Bubble sort

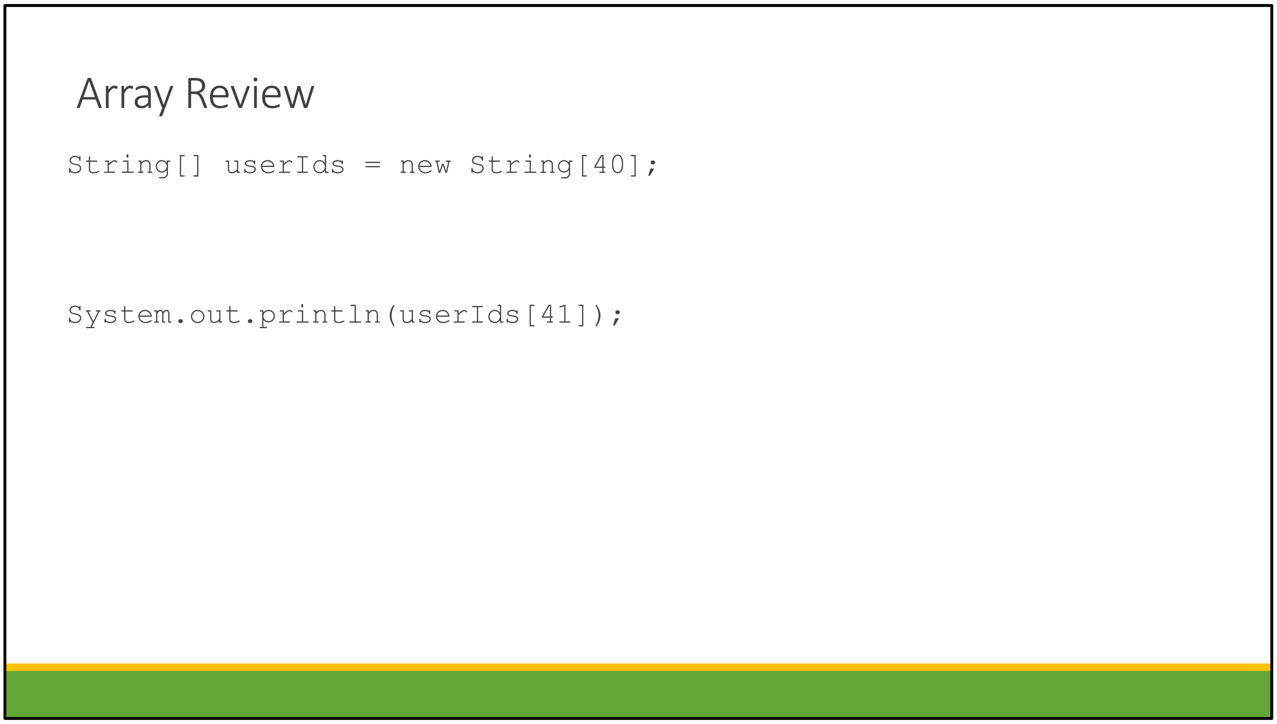


CS112 – Java Programming

Arrays and other Data Structures

Fall 2022

Start to study advanced data structures
What's a data structure? A structure for storing, retrieving, and manipulating data, with desired properties
Data, list, stack, tree, dictionary, etc



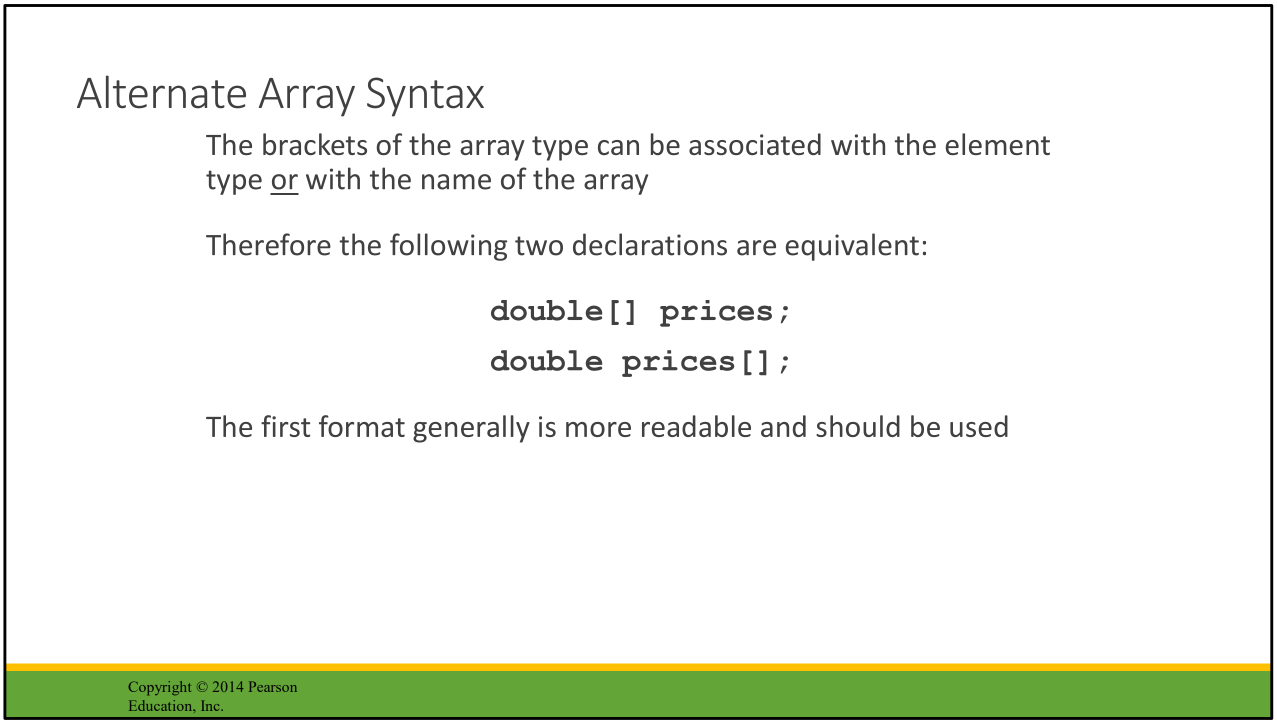
Array Review

```
String[] userIds = new String[40];



System.out.println(userIds[41]);
```

Constructed with new
Specify its length at creation
Must construct each element in the list (if not a basic type)
userIds.length is a member variable in the array
ArrayIndexOutOfBoundsException



Alternate Array Syntax

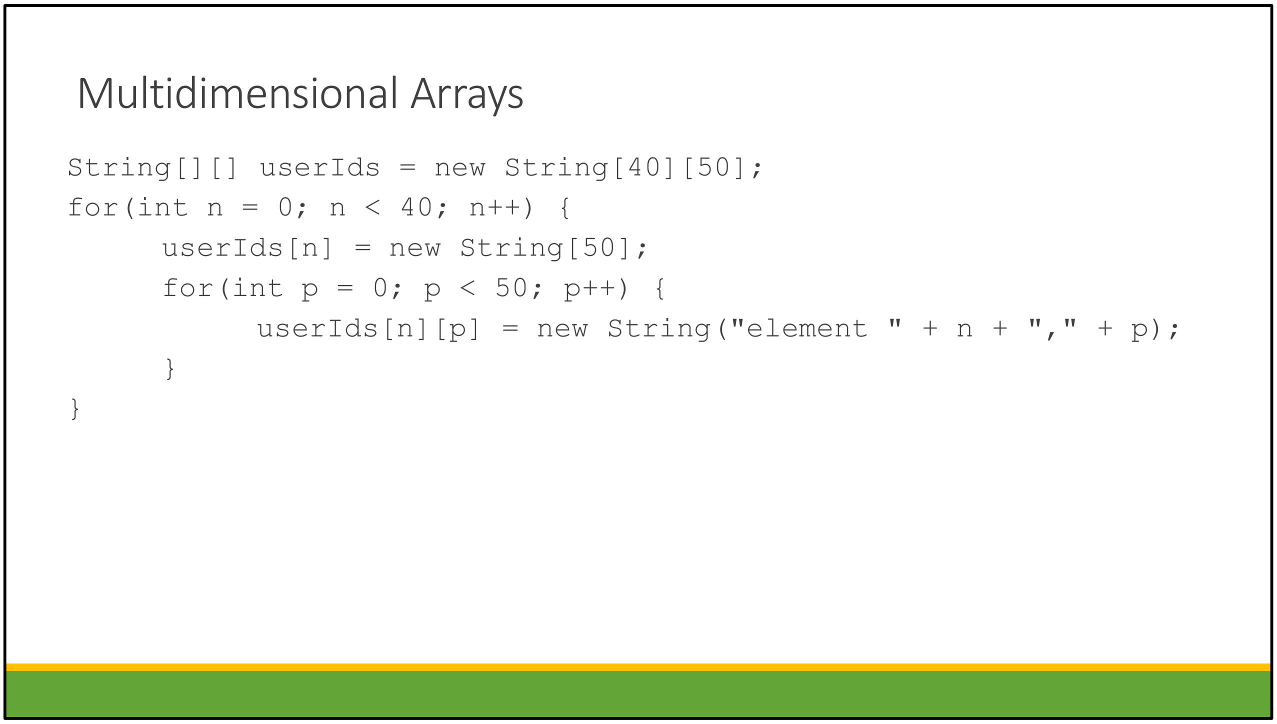
The brackets of the array type can be associated with the element type or with the name of the array

Therefore the following two declarations are equivalent:

```
double[] prices;
double prices[];
```

The first format generally is more readable and should be used

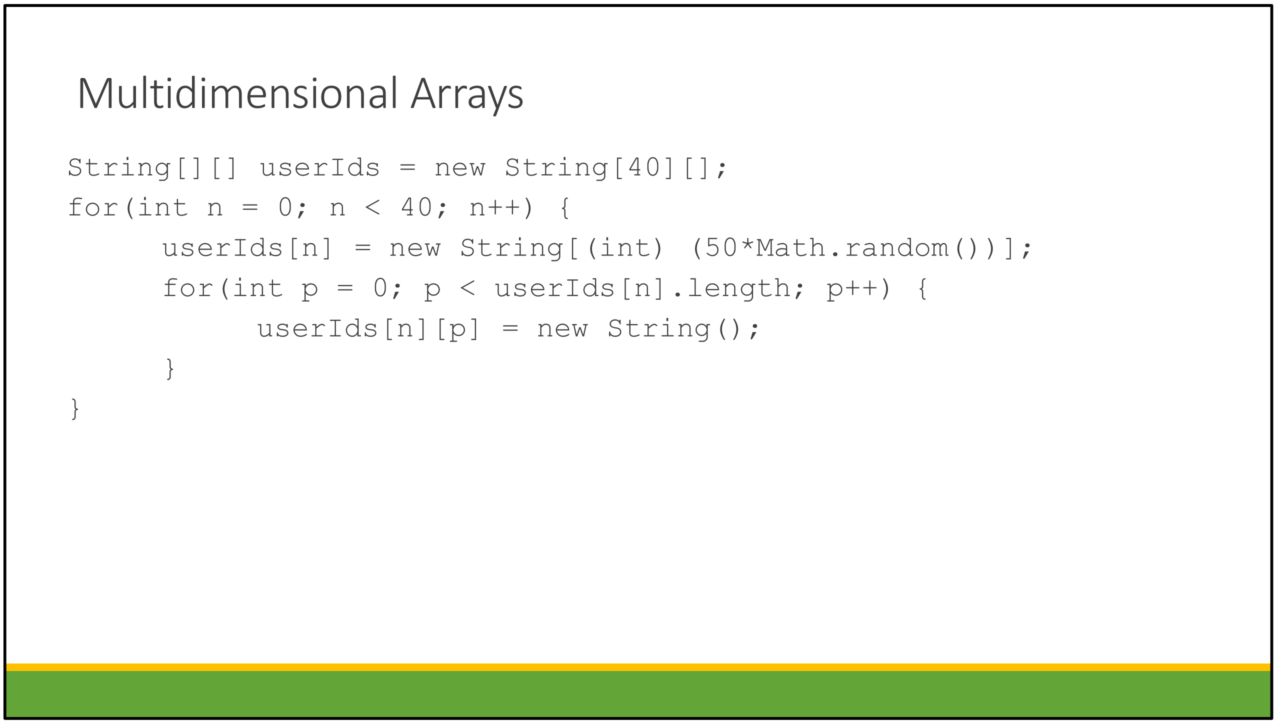
Hmm.



Multidimensional Arrays

```
String[][] userIds = new String[40][50];
for(int n = 0; n < 40; n++) {
      userIds[n] = new String[50];
      for(int p = 0; p < 50; p++) {
            userIds[n][p] = new String("element " + n + "," + p);
      }
}
```

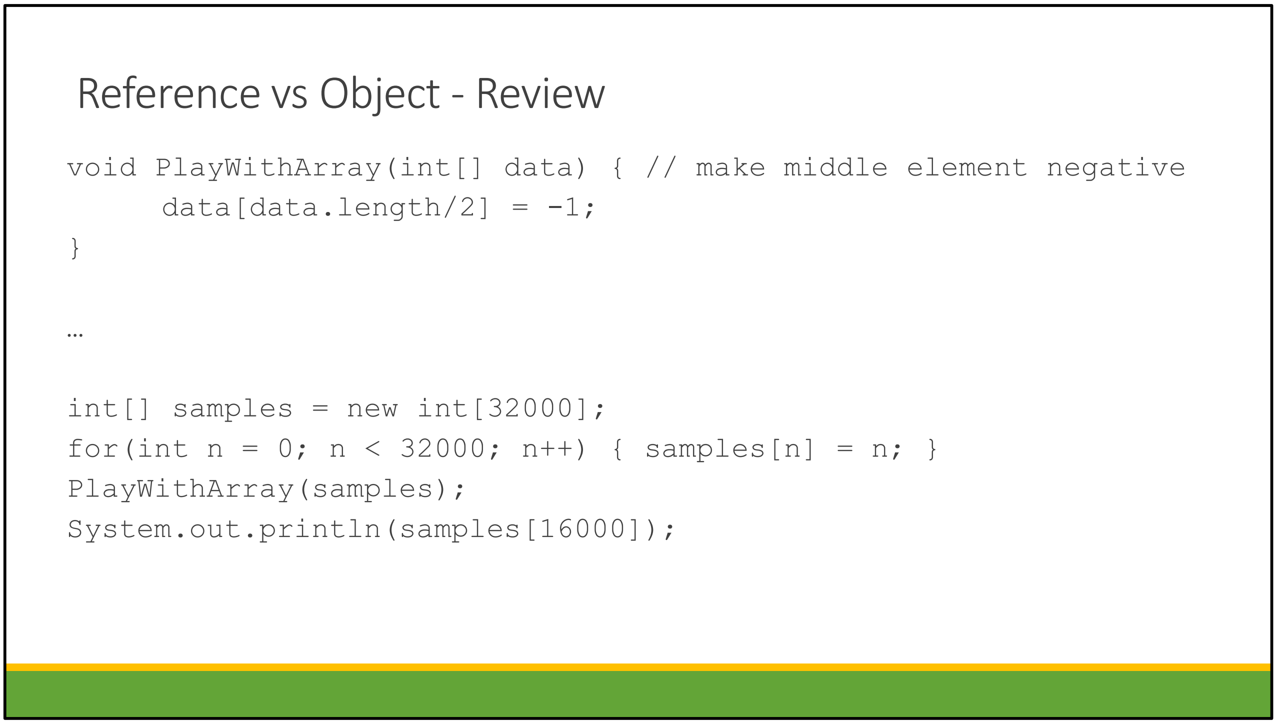
Is that ok?
What is the type of userIds?
What is the type of userIds[17]?
What is the type of userIds[17][3]?



Multidimensional Arrays

```
String[][] userIds = new String[40][];
for(int n = 0; n < 40; n++) {
     userIds[n] = new String[(int) (50*Math.random())];
     for(int p = 0; p < userIds[n].length; p++) {
          userIds[n][p] = new String();
     }
}
```

Is that ok?
YES. SubARRAYS NEED NOT ALL BE THE SAME LENGTH
Must specify at least first index
Must construct (with new) each subcomponent
Why would we use multidimensional arrays?
Video! Width, height, time, color. Audio! Time, left/right channel. Stock price! Time vs NYSE, S&P, NASDAQ, etc.



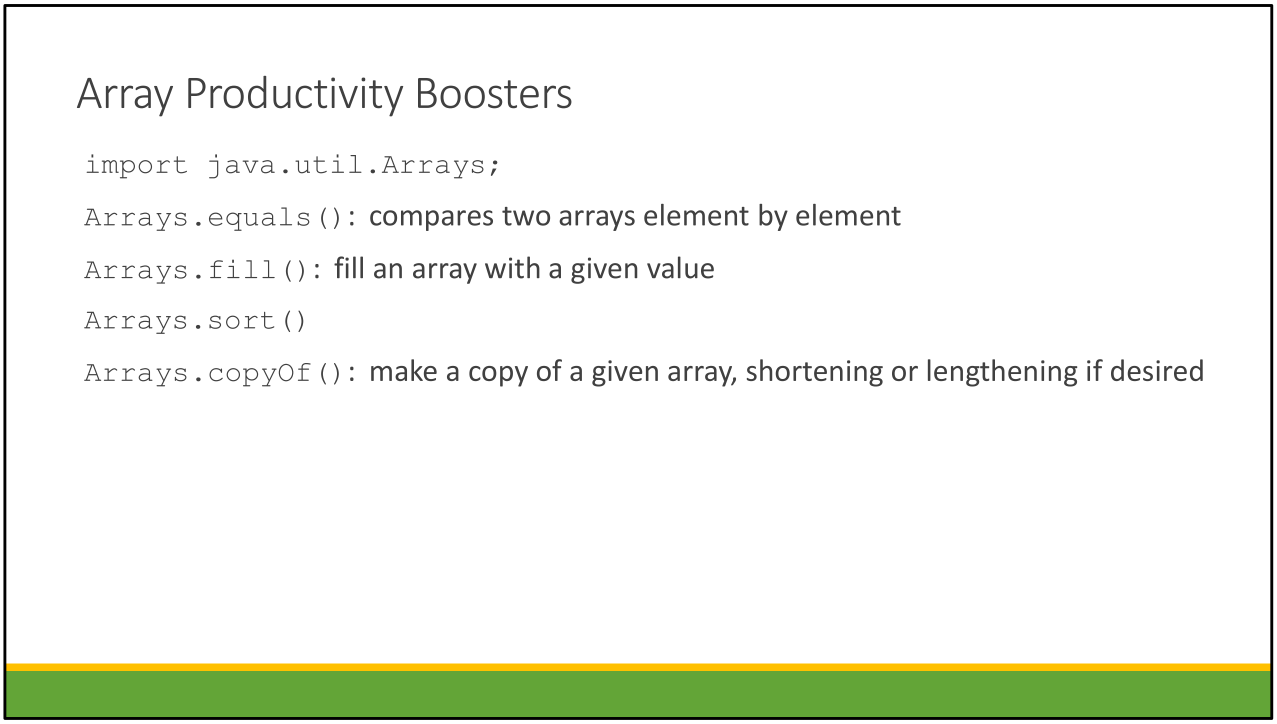
Reference vs Object - Review

```
void PlayWithArray(int[] data) { // make middle element negative
     data[data.length/2] = -1;
}

…

int[] samples = new int[32000];
for(int n = 0; n < 32000; n++) { samples[n] = n; }
PlayWithArray(samples);
System.out.println(samples[16000]);
```

What does this do? Prints 16k or -1? Try it out!
Does PlayWithArray change the value of samples outside of that function?
Yes!
Not immutable? CORRECT
Strings, Wrapper classes are immutable. We can build classes to be immutable or not. <u>But arrays are not immutable</u>.
Reference to array is passed. Referred-to array object CAN be modified by the reference passed inside to PlayWithArray



Array Productivity Boosters

`import java.util.Arrays;`

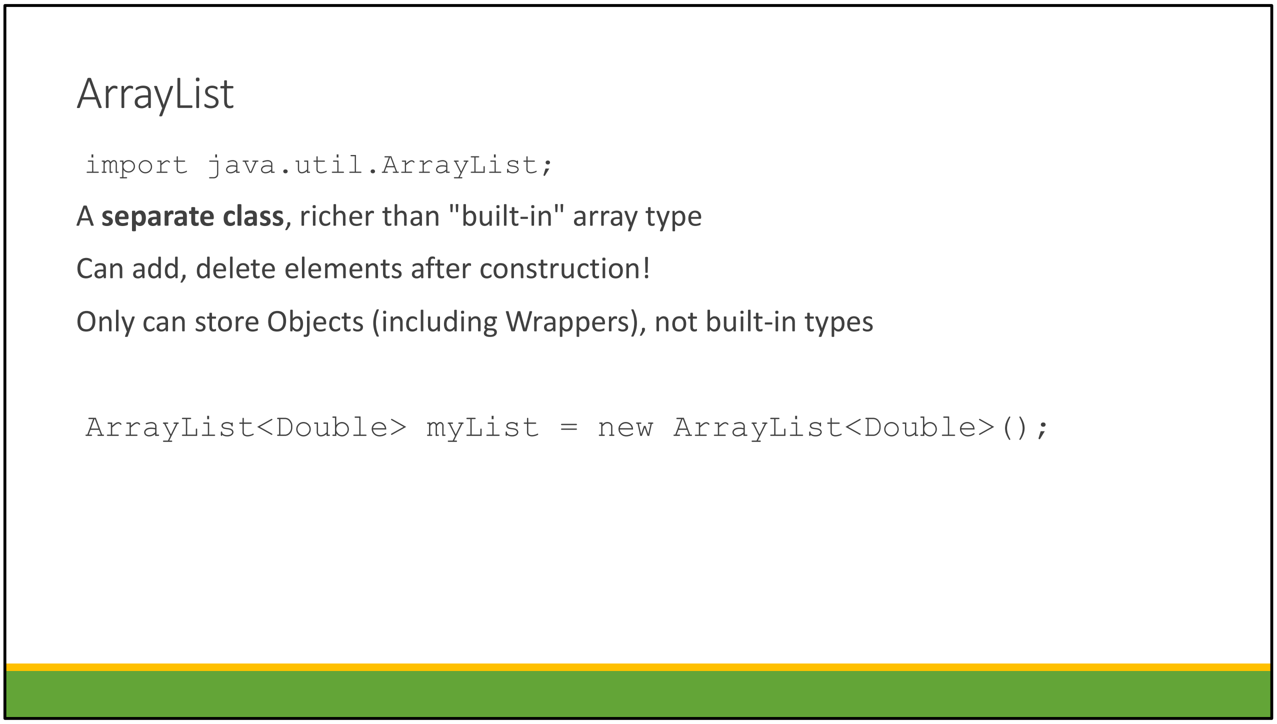
`Arrays.equals()`: compares two arrays element by element

`Arrays.fill()`: fill an array with a given value

`Arrays.sort()`

`Arrays.copyOf()`: make a copy of a given array, shortening or lengthening if desired

Lots of other methods also. Just FYI in case you want these



ArrayList

```
import java.util.ArrayList;
```

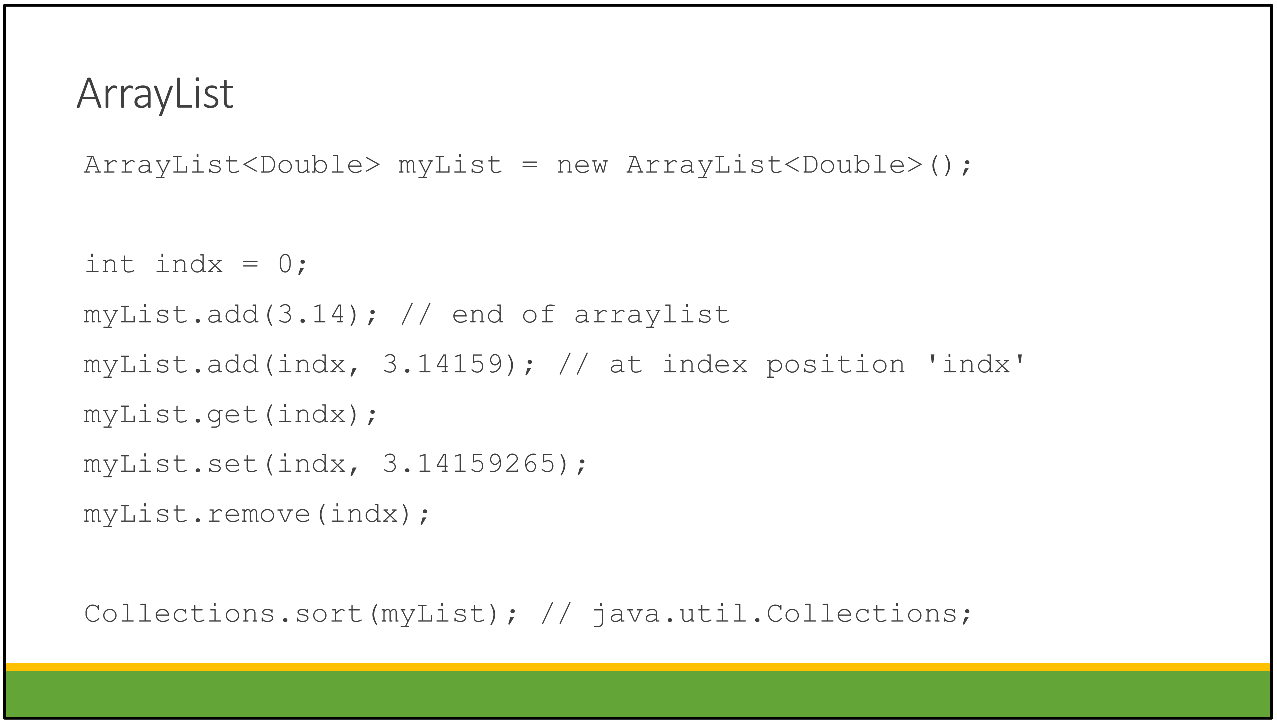
A **separate class**, richer than "built-in" array type

Can add, delete elements after construction!

Only can store Objects (including Wrappers), not built-in types

```
ArrayList<Double> myList = new ArrayList<Double>();
```

This is a new class.
NEW SYNTAX. ArrayList is a "generic“ (like a template) not an actual type. Add an Object in brackets to get the actual type
ArrayList<Double>, ArrayList<String>, etc are actual types.



ArrayList

```
ArrayList<Double> myList = new ArrayList<Double>();

int indx = 0;
myList.add(3.14); // end of arraylist
myList.add(indx, 3.14159); // at index position 'indx'
myList.get(indx);
myList.set(indx, 3.14159265);
myList.remove(indx);

Collections.sort(myList); // java.util.Collections;
```

Args are indices into the arraylist!

Plenty more methods: size(), methods to find index of a value, etc.

This is great! Why not use it all the time?
- No built-in types
- Less efficient than built-in arrays
- But very useful and used A LOT

```
//********************************************************************
//  Beatles.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a ArrayList object.
//********************************************************************

import java.util.ArrayList;

public class Beatles
{
   //-----------------------------------------------------------------
   //  Stores and modifies a list of band members.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      ArrayList<String> band = new ArrayList<String>();

      band.add("Paul");
      band.add("Pete");
      band.add("John");
      band.add("George");
```

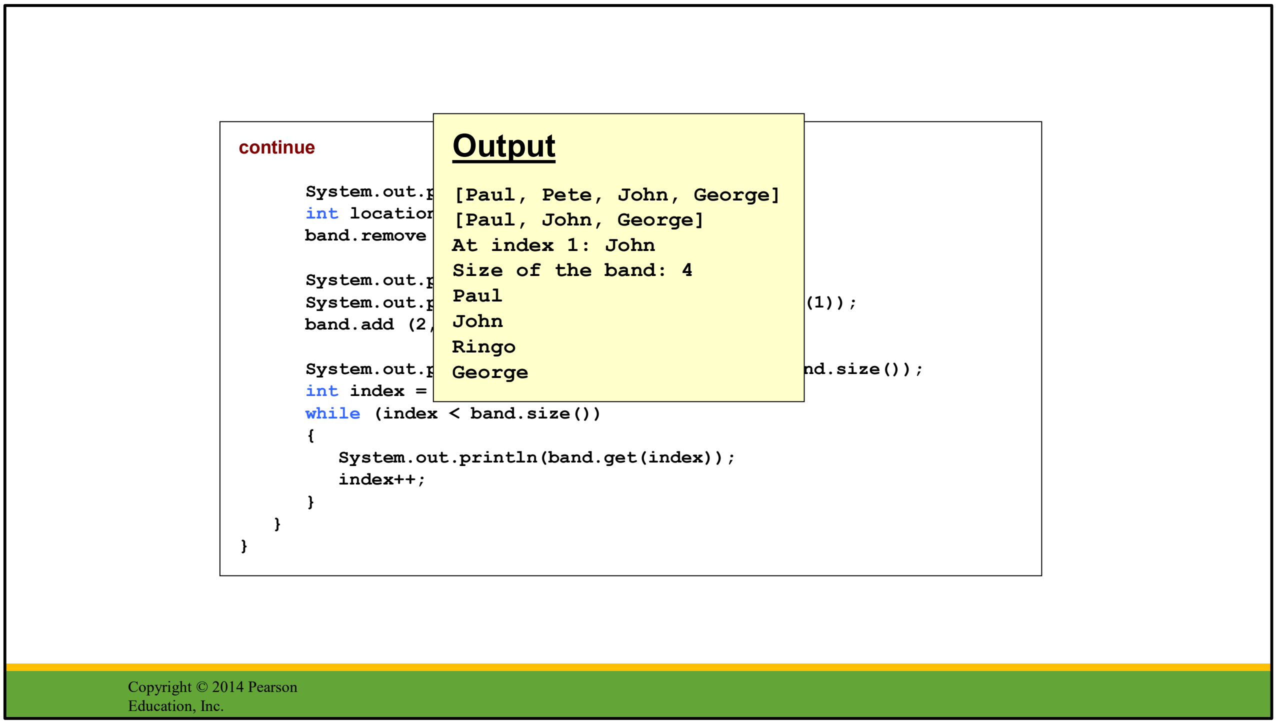
continue

continue

```
      System.out.println(band);
      int location = band.indexOf("Pete");
      band.remove(location);

      System.out.println(band);
      System.out.println("At index 1: " + band.get(1));
      band.add(2, "Ringo");

      System.out.println("Size of the band: " + band.size());
      int index = 0;
      while (index < band.size())
      {
         System.out.println(band.get(index));
         index++;
      }
   }
}
```



continue

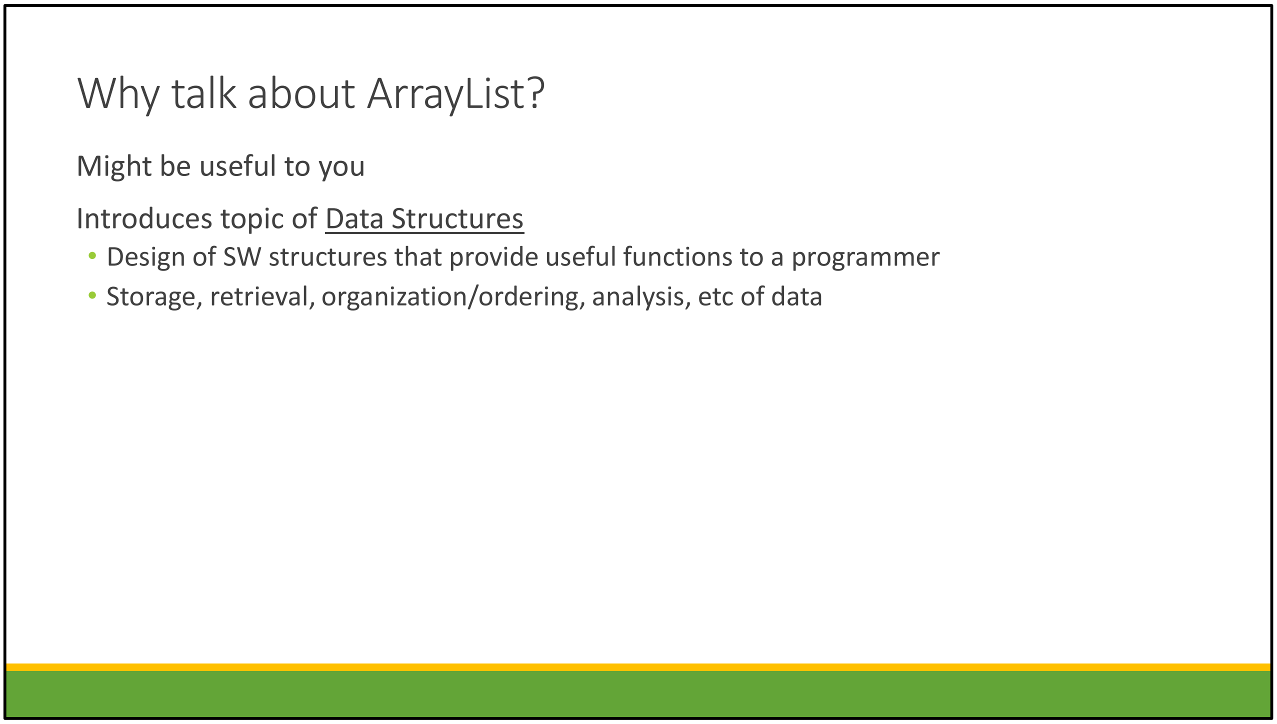
```
      System.out.p
      int locatior
      band.remove

      System.out.p
      System.out.p                                          (1));
      band.add (2,

      System.out.p                                         nd.size());
      int index =
      while (index < band.size())
      {
         System.out.println(band.get(index));
         index++;
      }
   }
}
```

Output

```
[Paul, Pete, John, George]
[Paul, John, George]
At index 1: John
Size of the band: 4
Paul
John
Ringo
George
```



Why talk about ArrayList?

Might be useful to you

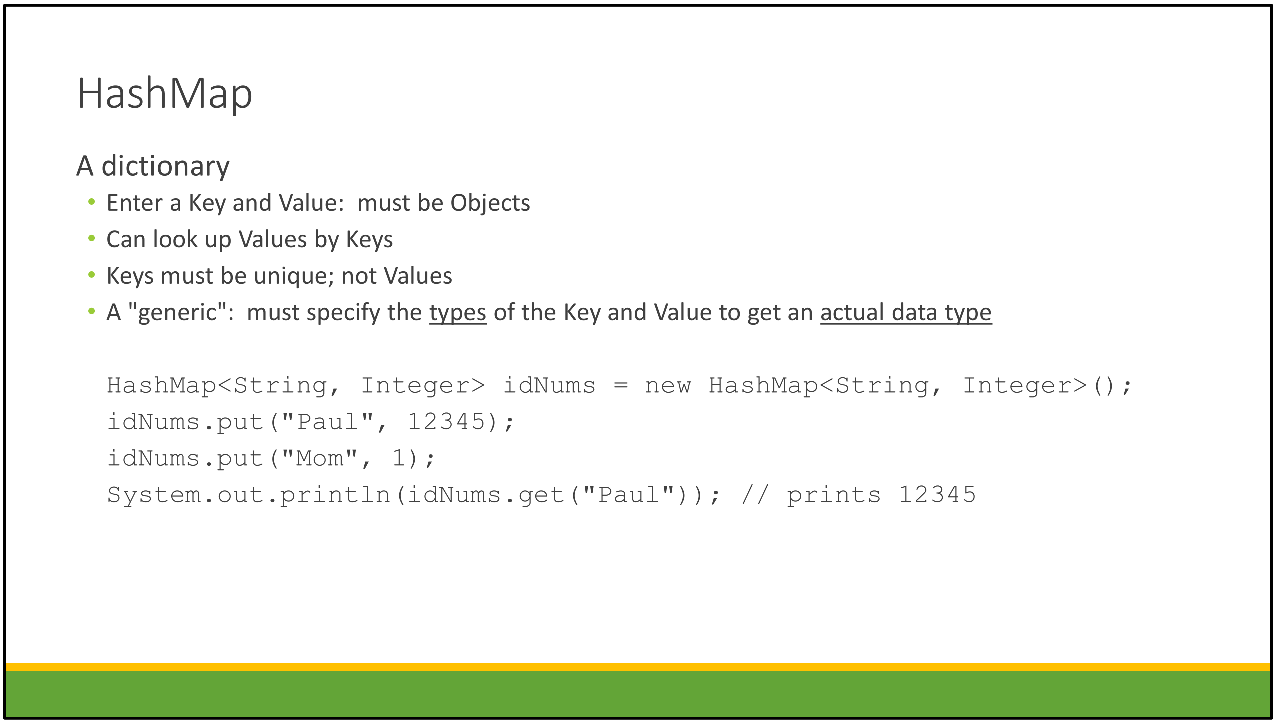
Introduces topic of Data Structures

- Design of SW structures that provide useful functions to a programmer
- Storage, retrieval, organization/ordering, analysis, etc of data

I expect you to learn & memorize all methods of ArrayList & their behavior?? NO

We will learn about, design, build, and test our own data structures
Whole courses on this topic

More Data Structures…

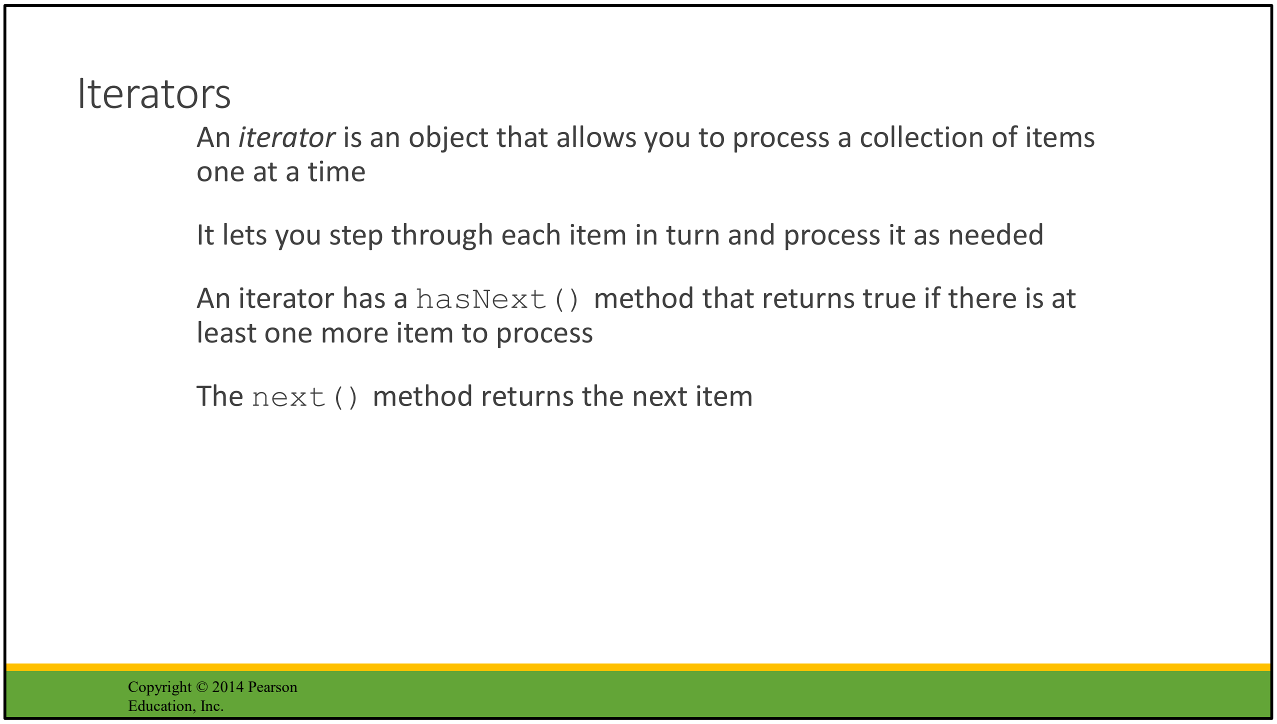


HashMap

A dictionary

- Enter a Key and Value: must be Objects
- Can look up Values by Keys
- Keys must be unique; not Values
- A "generic": must specify the types of the Key and Value to get an actual data type

```
HashMap<String, Integer> idNums = new HashMap<String, Integer>();
idNums.put("Paul", 12345);
idNums.put("Mom", 1);
System.out.println(idNums.get("Paul")); // prints 12345
```



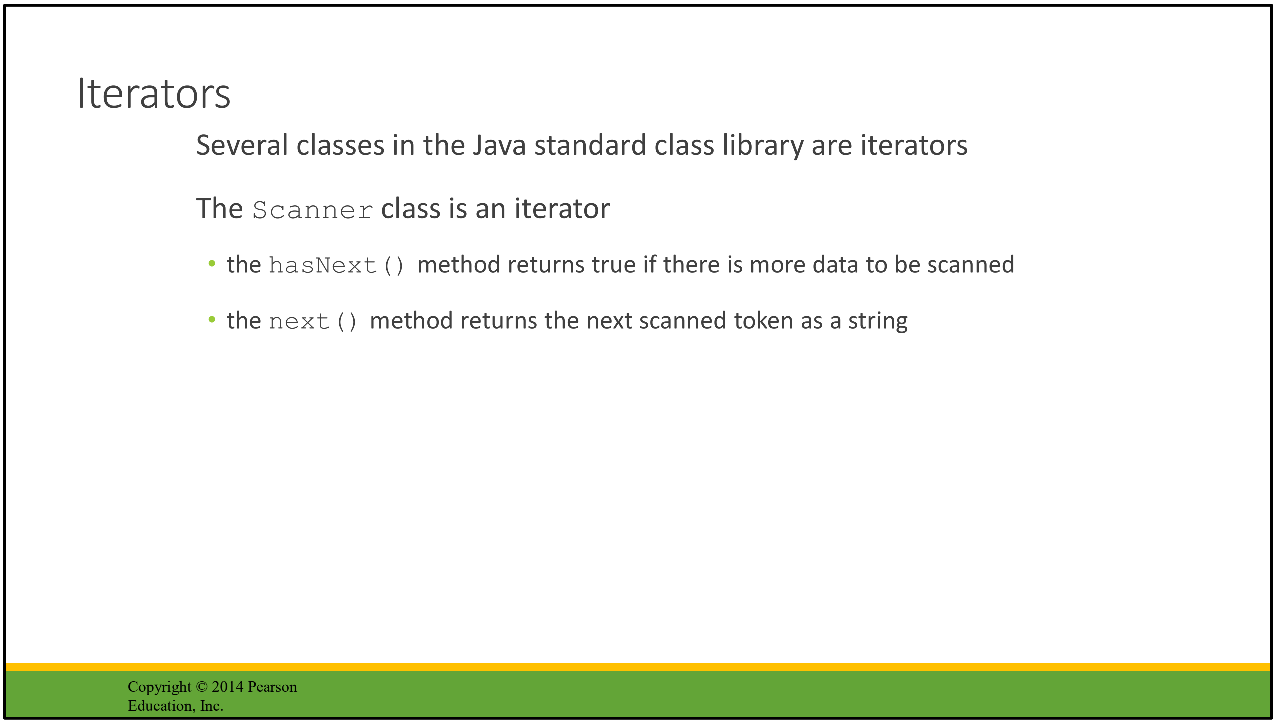
Iterators

An *iterator* is an object that allows you to process a collection of items one at a time

It lets you step through each item in turn and process it as needed

An iterator has a `hasNext()` method that returns true if there is at least one more item to process

The `next()` method returns the next item

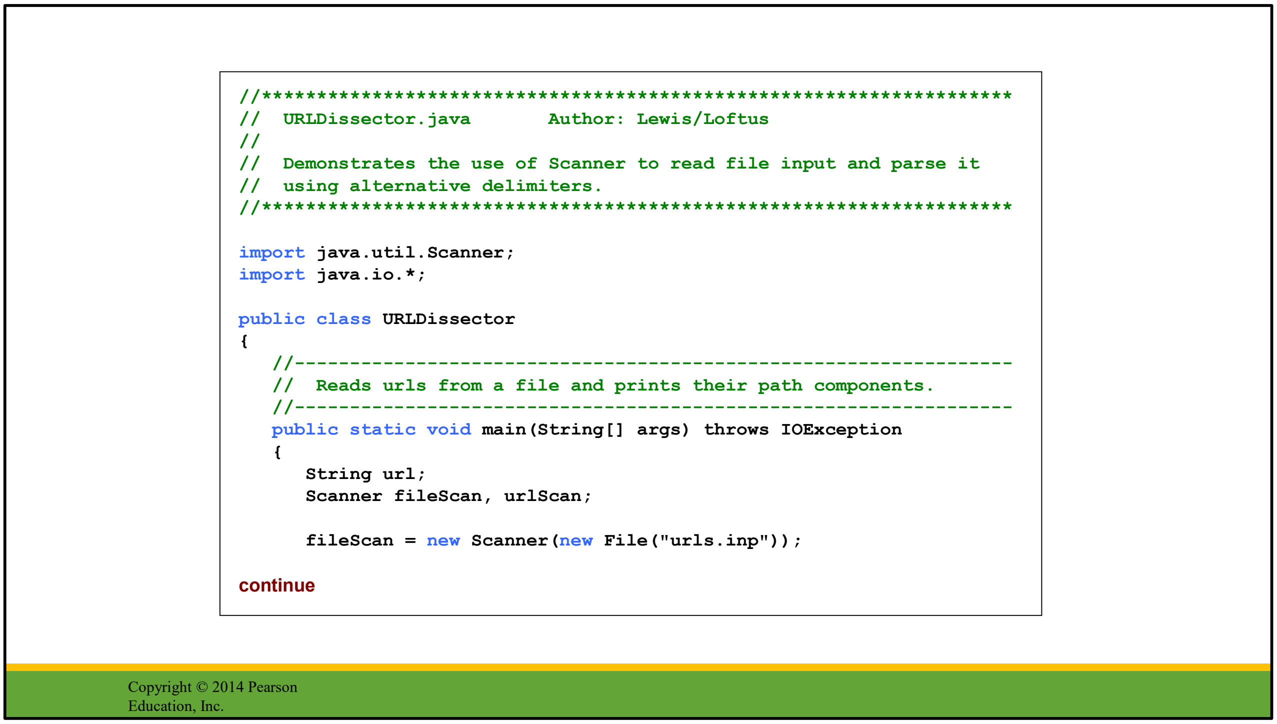


Iterators

Several classes in the Java standard class library are iterators

The `Scanner` class is an iterator

- the `hasNext()` method returns true if there is more data to be scanned
- the `next()` method returns the next scanned token as a string



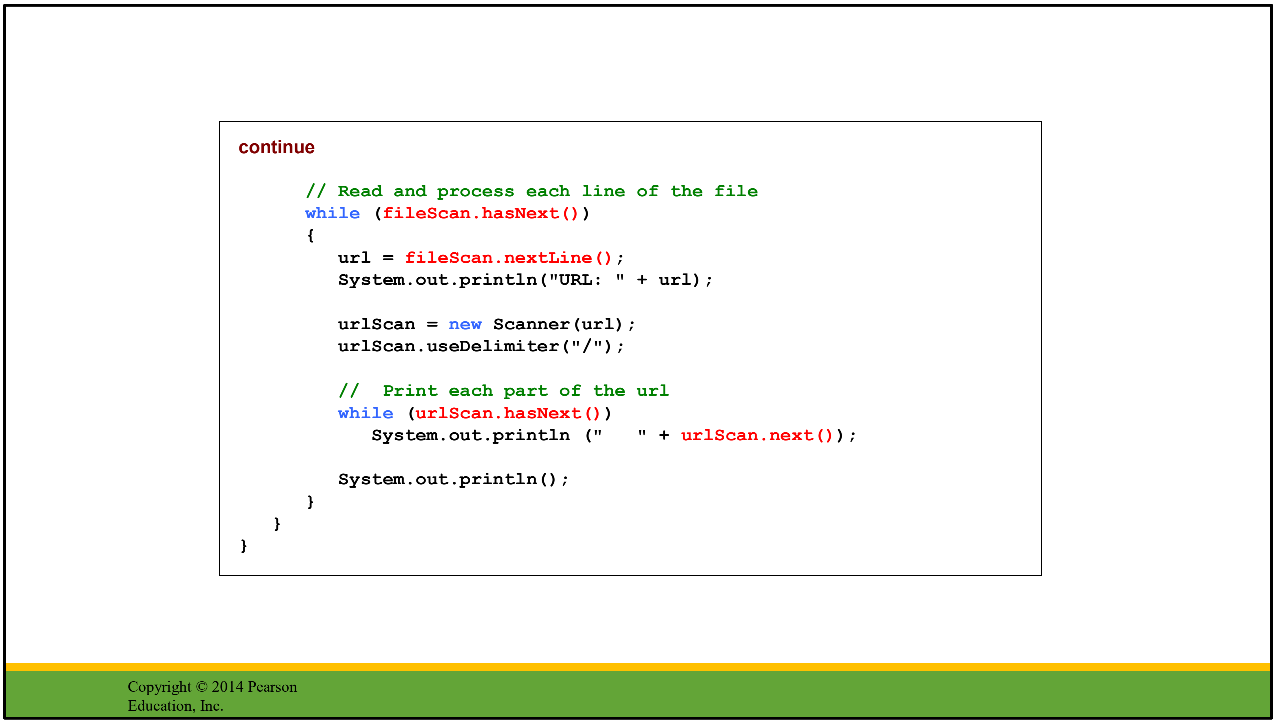
```
//********************************************************************
//  URLDissector.java       Author: Lewis/Loftus
//
//  Demonstrates the use of Scanner to read file input and parse it
//  using alternative delimiters.
//********************************************************************

import java.util.Scanner;
import java.io.*;

public class URLDissector
{
   //-----------------------------------------------------------------
   //  Reads urls from a file and prints their path components.
   //-----------------------------------------------------------------
   public static void main(String[] args) throws IOException
   {
      String url;
      Scanner fileScan, urlScan;

      fileScan = new Scanner(new File("urls.inp"));
```

continue



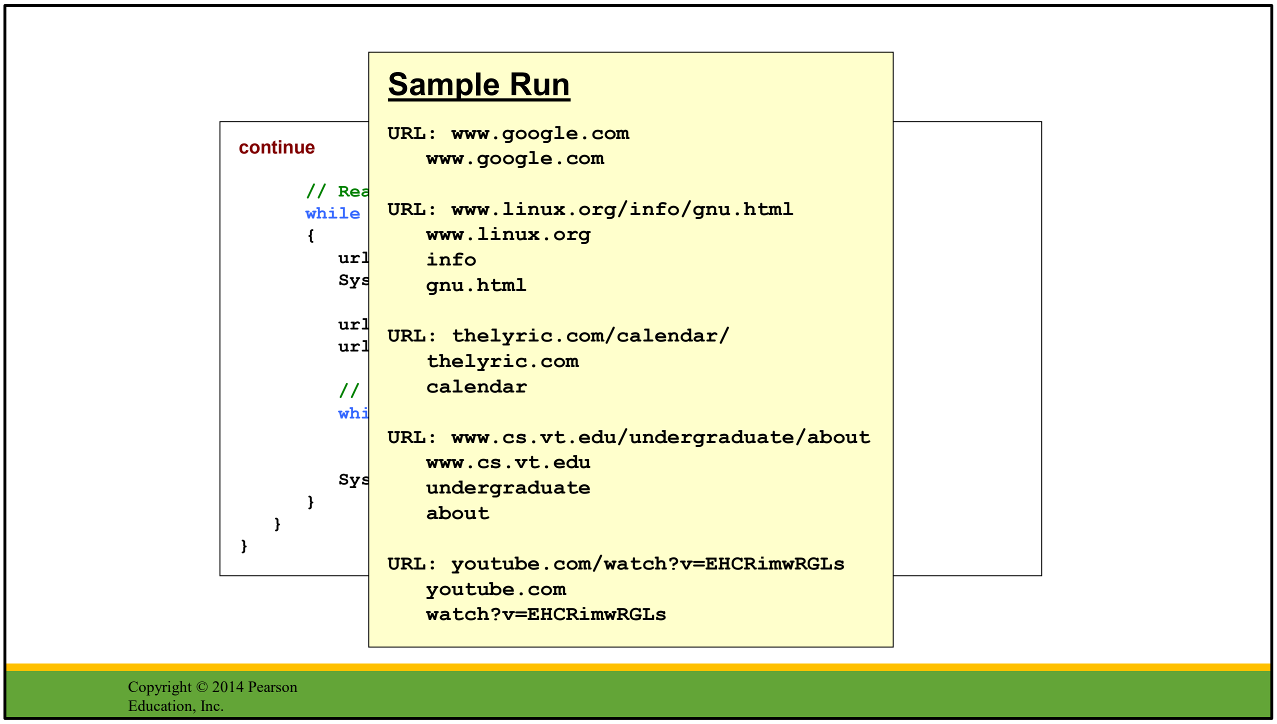
continue

```
      // Read and process each line of the file
      while (fileScan.hasNext())
      {
         url = fileScan.nextLine();
         System.out.println("URL: " + url);

         urlScan = new Scanner(url);
         urlScan.useDelimiter("/");

         //  Print each part of the url
         while (urlScan.hasNext())
            System.out.println ("   " + urlScan.next());

         System.out.println();
      }
   }
}
```



continue

```
        // Rea
        while
        {
            url
            Sys

            url
            url

            //
            whi


            Sys
        }
    }
}
```

Sample Run

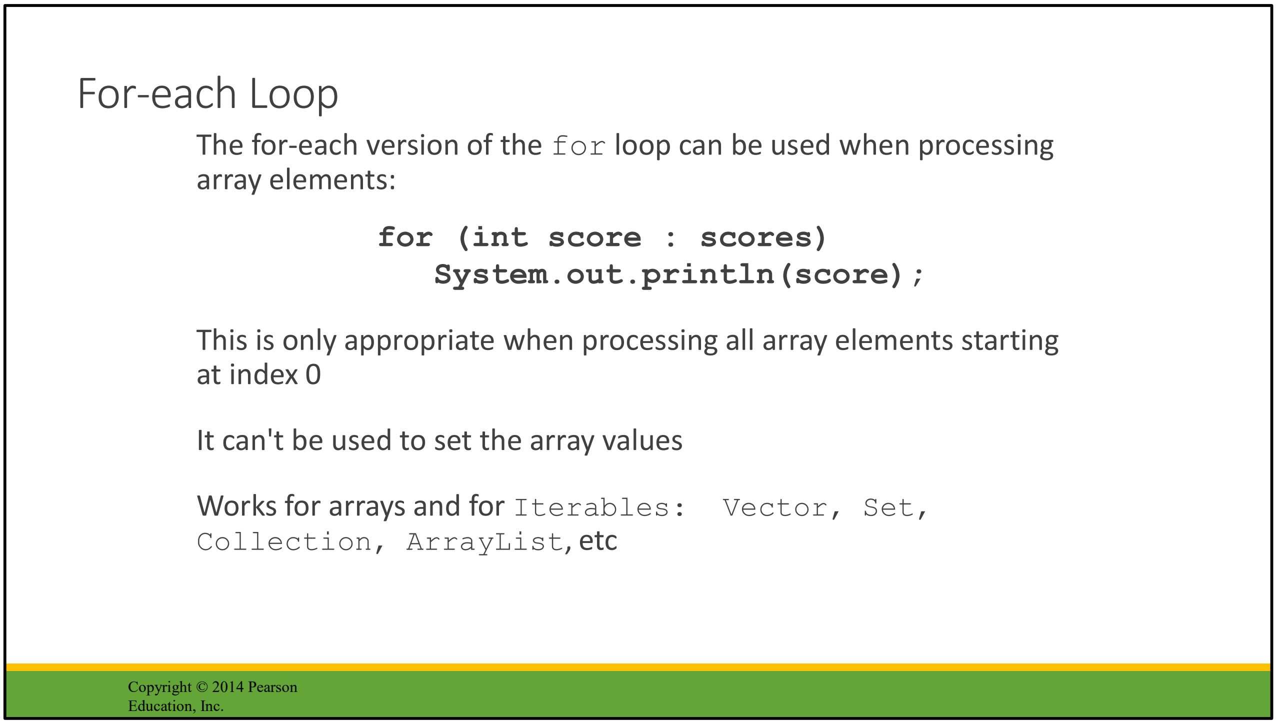
```
URL: www.google.com
   www.google.com

URL: www.linux.org/info/gnu.html
   www.linux.org
   info
   gnu.html

URL: thelyric.com/calendar/
   thelyric.com
   calendar

URL: www.cs.vt.edu/undergraduate/about
   www.cs.vt.edu
   undergraduate
   about

URL: youtube.com/watch?v=EHCRimwRGLs
   youtube.com
   watch?v=EHCRimwRGLs
```



For-each Loop

The for-each version of the `for` loop can be used when processing array elements:

```
for (int score : scores)
   System.out.println(score);
```

This is only appropriate when processing all array elements starting at index 0

It can't be used to set the array values

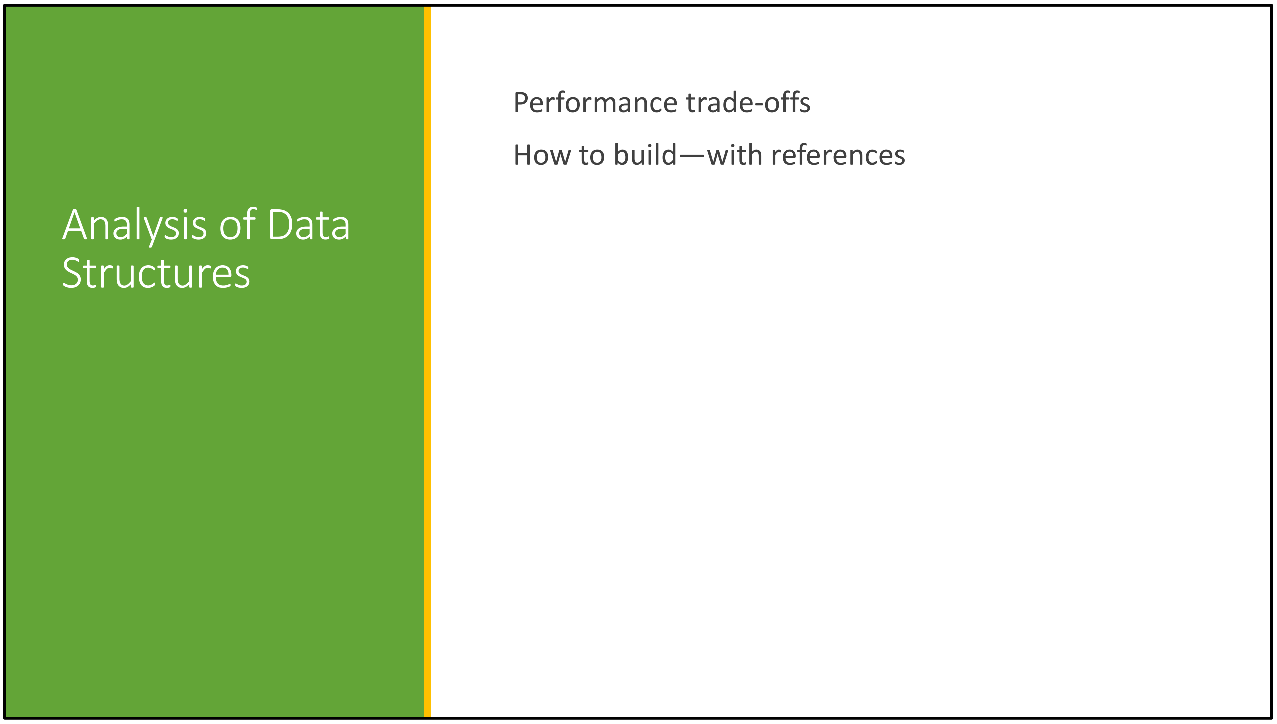
Works for arrays and for `Iterables:  Vector, Set, Collection, ArrayList,` etc

Like Python!
HashMap has related methods:

- Set<K> keySet()
- Collection<V> values()

DO EXAMPLE

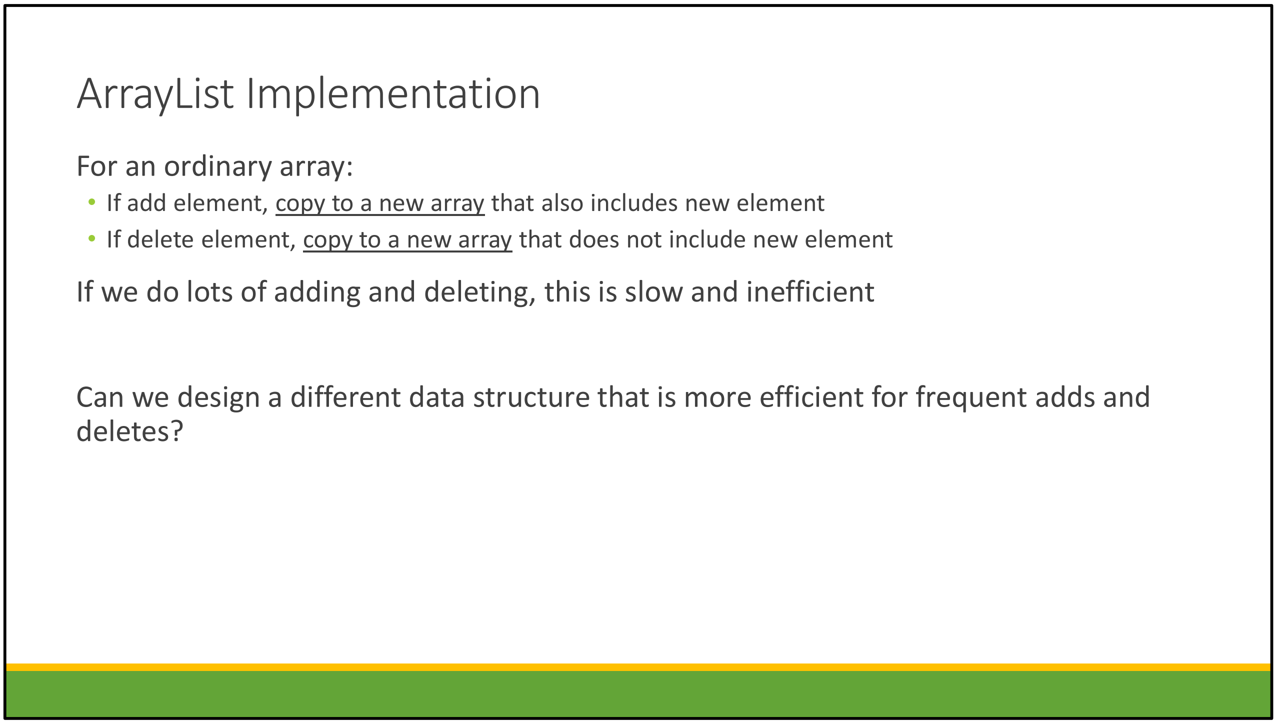
```
for (String x : map.keySet()) {
System.out.println("" + x + " : " + map.get(x));
}
```



Analysis of Data Structures

Performance trade-offs

How to build—with references



ArrayList Implementation

For an ordinary array:

- If add element, copy to a new array that also includes new element
- If delete element, copy to a new array that does not include new element

If we do lots of adding and deleting, this is slow and inefficient

Can we design a different data structure that is more efficient for frequent adds and deletes?

If we spend much more time looking up values than adding/deleting, ArrayList is great.